CS 4341 Report 3

Kewen Gu(kgu) & Zhaochen Ding(zding2)

Part I:

Feature Description

In this Project, we implemented 5 features as listed below:

1. Which Player has more pieces in center rows.
2. Which Player has a piece at the bottom left corner of the board.
3. Which Player has more pieces in the center column.
4. Which Player has more pieces in the bottom row.
5. Which Player has more connected 2 piece in a row.

The output of these five features are some small integers, either positive, 0, or negative. If the player 1 has more advantage over player 2, the output is positive; if both player has the same advantage, the output is 0; otherwise, the output is negative.

Feature Design Strategy

The first two features are the required feature; we based on this and considered the 3rd and 4th feature, since we think the center column and the bottom row are both important places since the player who has more piece in these two places are more likely to form a connection in a step even if the player is going to stop an enemy connection from forming. The 5th feature is pretty straightforward, the player who has more 2 connected piece will be more likely to form several 3 piece connections that are hard for opponent to deal with, so it should be a good feature. We discussed about 3 connected piece but we consider this too hard to achieve, we don't want to see plenty of tied results.

Part II:

Decision Tree Implementation

We implemented a method that collects data from the given dataset and calculates entropy, as well as information gain of each feature. This algorithm will eventually generate a decision tree based on the information gains obtained from the data.

In addition, we used k-fold cross validation for implementing the decision tree. We first divide the data set read from file into k subsets. Then, we perform decision tree learning algorithm on k-1 sets of the data, and test the result decision tree classifiers using the

remaining 1 set. We repeat the process for k times, and calculated the average error rate from the classification results.

We have printed out the results of our program when the specified number of folds is 10, which is shown in figure below.

```
C:\Python27\python2.exe "C:/Users/Kewen Gu/PycharmProjects/AI Project3 Part2/Main.py" 10
*****************************************
Feature 1
# of correctly classified instances: 82
# of incorrectly classified instances: 17
Error rate: 0.1717
*****************************************
Feature 2
# of correctly classified instances: 90
# of incorrectly classified instances: 9
Error rate: 0.0909
*****************************************
Feature 3
# of correctly classified instances: 78
# of incorrectly classified instances: 22
Error rate: 0.2200
*****************************************
Feature 4
# of correctly classified instances: 84
# of incorrectly classified instances: 15
Error rate: 0.1515
*****************************************
Feature 5
# of correctly classified instances: 76
# of incorrectly classified instances: 23
Error rate: 0.2323

Process finished with exit code 0
```

Decision Tree and Feature Performance

Surprisingly, our results shows that the second feature, which Player has a piece at the bottom left corner of the board, turns out to have the least error rate. We don't know the reason or logic behind this result, since we used to consider feature 2 to be the most irrelavant case toward determining the winner. The error rates for the other four features seem reasonable.

Part III:

Improvement in Heuristics

We added code that checks the return value of the five features for each board state after player drops a disc, and adds more points to its heuristic evaluation. If the result returned by each feature function is positive, we'll add multiply the result with a weight, and add to the heuristic score. Otherwise, we'll subtract the multiplication of result with weight from the heuristic. The weight is defined based on the correctness of its prediction of the final win. Then, based on the heuristic evaluation score, we do alpha-beta pruning on the possible board states, and pick the best move for the player.

Part IV:

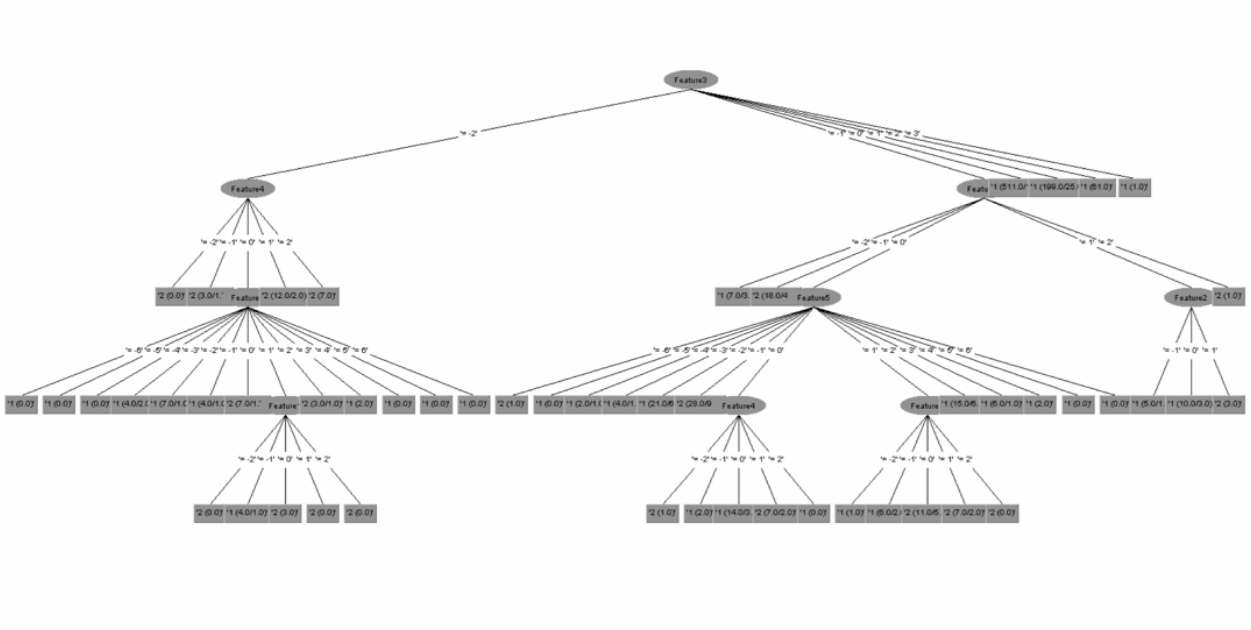
Extra Works - Weka

We used Weka to generate the decision tree and compared the error rates of generated decision tree with ours. We first filtered the dataset using NumericToNominal under unsupervised learning. Then, we chose J48 under trees as the classifier and ran the test with k-fold cross validation where K = 10, Weka returned an error rate of 24.7 %. Here is the screenshot of Weka output:

Classifier output

```
|   |   |   Feature4 = 0: 2 (11.0/5.0)
|   |   |   Feature4 = 1: 2 (7.0/2.0)
|   |   |   Feature4 = 2: 2 (0.0)
|   |   Feature5 = 2: 1 (15.0/6.0)
|   |   Feature5 = 3: 1 (6.0/1.0)
|   |   Feature5 = 4: 1 (2.0)
|   |   Feature5 = 5: 1 (0.0)
|   |   Feature5 = 6: 1 (0.0)
|   Feature1 = 1
|   |   Feature2 = -1: 1 (5.0/1.0)
|   |   Feature2 = 0: 1 (10.0/3.0)
|   |   Feature2 = 1: 2 (3.0)
|   Feature1 = 2: 2 (1.0)
Feature3 = 0: 1 (511.0/121.0)
Feature3 = 1: 1 (199.0/25.0)
Feature3 = 2: 1 (61.0)
Feature3 = 3: 1 (1.0)

Number of Leaves  :     52

Size of the tree :      61


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         753               75.3    %
Incorrectly Classified Instances       247               24.7    %
Kappa statistic                          0.2278
Mean absolute error                      0.3353
Root mean squared error                  0.4253
Relative absolute error                 86.4362 %
Root relative squared error             96.589  %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.929     0.741      0.778     0.929     0.847      0.67     1
                 0.259     0.071      0.567     0.259     0.355      0.67     2
Weighted Avg.    0.753     0.565      0.723     0.753     0.718      0.67

=== Confusion Matrix ===

   a   b   <-- classified as
 685  52 |   a = 1
 195  68 |   b = 2
```

And the decision tree generated by Weka looks like this:



Then we tried single predictability and removed four of the Features and run classification. Since we always want our target label class to be winner, we did the the Undo and deletion four times. However, all the outcomes are the same as figures shown below:

```
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     outputsFromPart1-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters
Instances:    1000
Attributes:   2
              Feature1
              winner
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
: 1 (1000.0/263.0)

Number of Leaves  :     1

Size of the tree :      1


Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         737               73.7    %
Incorrectly Classified Instances       263               26.3    %
Kappa statistic                          0
Mean absolute error                      0.3877
Root mean squared error                  0.4403
Relative absolute error                 99.9358 %
Root relative squared error             99.9999 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         1          0.737     1         0.849      0.495    1
                 0         0          0         0         0          0.495    2
Weighted Avg.    0.737     0.737      0.543     0.737     0.625      0.495

=== Confusion Matrix ===

   a   b   <-- classified as
 737   0 |   a = 1
 263   0 |   b = 2
```

```
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     outputsFromPart1-weka.filters.unsupervised.attribute.NumericToNominal-I
Instances:    1000
Attributes:   2
              Feature2
              winner
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
: 1 (1000.0/263.0)

Number of Leaves  :     1

Size of the tree :      1


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         737               73.7    %
Incorrectly Classified Instances       263               26.3    %
Kappa statistic                          0
Mean absolute error                      0.3877
Root mean squared error                  0.4403
Relative absolute error                 99.9358 %
Root relative squared error             99.9999 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         1          0.737     1         0.849      0.495    1
                 0         0          0         0         0          0.495    2
Weighted Avg.    0.737     0.737      0.543     0.737     0.625      0.495

=== Confusion Matrix ===

   a   b   <-- classified as
 737   0 |   a = 1
 263   0 |   b = 2
```

The other three features also looked same as the above picture.